

# A parallel fast multipole method for elliptic difference equations

S. Liska\*, T. Colonius

*Division of Engineering and Applied Science, California Institute of Technology,  
Pasadena, CA 91125, USA*

---

## Abstract

A new fast multipole formulation for solving elliptic PDEs on unbounded domains and its parallel implementation are presented. This method formally discretizes the PDE on an infinite Cartesian grid, and then solves the corresponding difference equations. In the analog to solving continuous inhomogeneous differential equations using Green's functions, the proposed method uses the fundamental solution of the discrete operator on an infinite grid, or lattice Green's function. Fast solutions  $\mathcal{O}(N)$  are achieved by using a kernel-independent interpolation-based fast multipole method. Unlike other fast multipole algorithms, our approach exploits the regularity of the underlying Cartesian grid and the efficiency of FFTs to reduce the computation time. Our parallel implementation allows communications and computations to be overlapped and requires minimal global synchronization. The accuracy, efficiency, and parallel performance of the method are demonstrated through numerical experiments on the discrete 3D Poisson equation.

*Keywords:* fast multipole method, fast convolution, difference equation, Green's function, infinite domain, parallel computing, discrete operator, elliptic solver

---



---

\*Corresponding author

*Email addresses:* `sliska@caltech.edu` (S. Liska), `colonius@caltech.edu` (T. Colonius)

## 1. Introduction

Numerical simulations of physical phenomena often require fast solutions to linear, elliptic PDEs with constant coefficients on unbounded domains. Grid-based methods typically approximate free-space boundary conditions by large computational domains and imposing approximate Dirichlet or Neuman boundary conditions. Such techniques increase the number of computational elements and often employ solvers that are less efficient than those used on regular grids (FFT techniques, multigrid, etc.). In contrast, many particle-methods have efficient nodal distributions and are able to satisfy free-space boundary conditions automatically. Recent developments in kernel-independent fast multipole methods (FMMs) [1, 2, 3, 4, 5, 6] have allowed such particle-methods to achieve fast solutions for a wide range of physical problems. The present method attempts to combine some attractive features of grid-based methods (FFT techniques, efficient data structures, etc.) with particle-methods (efficient nodal distribution, fast free-space solvers, etc.) in order to achieve fast solutions for system of difference equations on unbounded domains.

Our formulation formally discretizes the PDE on an infinite Cartesian grid. The infinite grid is then made finite by removing cells that contain negligible source strength. Solutions to the difference equations are obtained through the convolution of the fundamental solution of the discrete operator with the source terms of the difference equation [7, 8, 9, 6]. The fundamental solution of the various discrete operators, often referred to as lattice Green's functions (LGFs), have been studied in various fields, such as random walks [10], crystal physics [11], modeling in quantum physics [12], and numerical analysis [13]. Although LGFs have been extensively studied, they have rarely been used for solving large systems of inhomogeneous difference equations (exceptions include 2D problems [14, 9, 6]).

We are particularly interested in using fast LGF algorithms to solve difference equations obtained from numerical discretization schemes that enforce discrete conservation laws. In such cases, physical fidelity is achieved through accurate solutions to the discrete equations, and not necessarily the original PDE. Examples of these discretization techniques include finite-volume methods, mimetic schemes, covolume methods, and discrete calculus methods. It is beyond the scope of this paper to provide a detailed description of these methods and their applications, instead we refer the reader to the recent review article [15] and the works included in [16].

Expressions for LGFs can be readily obtained in the form of Fourier integrals, but it is generally difficult to reduce the integral representations to expressions only involving a few elementary functions [17, 18]. Procedures for obtaining the asymptotic expansions of LGFs have been developed in [19, 20, 8]. Despite the availability of asymptotic expansions, such expressions are typically more complicated than the Green’s functions of their analogous continuous operators.

Solving the system of difference equations using LGFs requires evaluating discrete convolutions of the form

$$u(\mathbf{x}_i) = [K * \mathbf{f}](\mathbf{x}_i) = \sum_{j=0}^{M-1} K(\mathbf{x}_i, \mathbf{y}_j) f(\mathbf{y}_j), \quad i = 0, 1, \dots, N-1, \quad (1)$$

where  $K(\mathbf{x}_i, \mathbf{y}_j)$  is the kernel describing the influence of a source located at  $\mathbf{y}_j$  with strength  $f(\mathbf{y}_j)$  has on the field  $u(\mathbf{x})$  at location  $\mathbf{x}_i$ . For the case of  $M = N$ , the straightforward approach to evaluate Eq. 1 requires  $\mathcal{O}(N^2)$  operations. There are several techniques for evaluating Eq. 1 in  $\mathcal{O}(N)$  or  $\mathcal{O}(N \log N)$  operations. A few of these techniques are FMMs, FFT-based methods, particle-in-cell methods, particle-mesh methods, multigrid techniques, multilevel local-correction methods, and hierarchical matrix techniques. In the interest of brevity, a literature review of all the methods related to the fast evaluation of Eq. 1 is omitted; instead we focus our attention on FMMs.

The performance of FMMs relies on the existence of a compressed, or low-rank, representation of the far-field behavior of  $K(\mathbf{x}, \mathbf{y})$  that can be used to evaluate Eq. 1 to a prescribed tolerance. Classical fast multipole methods [21, 22] require analytical expansions of the far-field behavior of kernels in order to derived low-rank approximations. Although classical FMMs can be developed for the asymptotic expansion of LGFs, alternative FMMs that are better suited for complicated kernel expressions have been developed. Kernel-independent FMMs [1, 2, 3, 4, 5, 6] do not require analytical expansions of the far-field; instead, for suitable kernels, these methods only require numerical evaluations of the kernel.

The present method is an extensions of kernel-independent interpolation-based FMMs [23, 4], suitable for non-oscillatory translation-invariant kernels. Interpolation-based FMMs achieve low-rank approximations by projecting the kernel onto a finite basis of interpolation functions. The methods proposed in [23, 4] use Chebyshev interpolation and accelerate convolutions in-

volving the compressed kernel using a singular-value-decomposition (SVD). In contrast, our method uses polynomial interpolation on equidistant nodes to obtain a low-rank representation of the kernel, and accelerate convolutions involving the compressed kernel using FFTs. The use of intermediate regular grids and fast FFT-based convolutions has been used by other FMMs [24, 25, 2], and shown to be particularly useful in accelerating the computations of 3D methods [2]. The use of intermediate regular grids in our method has the added advantage of simplifying the multilevel algorithm, since sources and evaluation points are defined on Cartesian grids at all levels of the multilevel scheme. The spatial regularity allows for same fast convolution techniques to be used in determining near-field and far-field contributions. In addition to the base algorithm, our method allows for pre-computations that further accelerate the solver.

A 2D FMM for LGFs has been recently introduced by Gillman and Martinsson [6]. Aside from its relevance to LGFs, we find that the method presented in [6] to be inherently different from our method. Similar reductions in the operation count that we achieve through the combined use of polynomial interpolation and convolution via FFTs are achieved in [6] through the combined use of skeleton/proxy points and rank-revealing factorizations. Although we think it is possible to extend the FMM of [6] to 3D, we refrain from speculating on the performance of the method since such extensions are not explored in present literature and their details are unclear to us.

Details regarding LGFs and their relation to solving PDEs on unbounded domains are presented in Section 2. This section also describes methods for performing fast convolutions based on kernel compression and FFT techniques, and presents a context in which these two techniques can be combined to yield an even faster convolution scheme. The resulting fast multipole algorithm and its parallel extension are then described in Section 3. Finally, serial and parallel numerical experiments are reported and analyzed in Section 4.

## 2. Lattice Green's functions and fast block-wise convolution techniques

### 2.1. Solving difference equations on infinite Cartesian grids

The method proposed in this paper is designed to solve various inhomogeneous, linear, constant-coefficient PDEs on unbounded domains. As a representative problem, we consider in detail Poisson's equation in 3D,

$$[\Delta u](\mathbf{x}) = f(\mathbf{x}), \quad \text{supp}(f) \in \Omega, \quad (2)$$

where  $\mathbf{x} \in \mathbb{R}^3$ ,  $\Omega$  is a bounded domain in  $\mathbb{R}^3$ , and  $u(\mathbf{x})$  decays as  $1/|\mathbf{x}|$  at infinity. Equation 2 has the analytic solution

$$u(\mathbf{x}) = [G * f](\mathbf{x}) = \int_{\Omega} G(\mathbf{x} - \boldsymbol{\xi}) f(\boldsymbol{\xi}) d\boldsymbol{\xi}, \quad (3)$$

where  $G(\mathbf{x}) = -1/4\pi|\mathbf{x}|$  is the fundamental solution of the Laplace operator. Discretizing Eq. 2 on an infinite uniform Cartesian grid using a standard second-order finite-difference or finite-volume scheme produces a set of difference equations

$$[\mathbf{L}u](\mathbf{n}) = f(\mathbf{n}), \quad \text{supp}(f) \in \Omega_h, \quad (4)$$

where  $\mathbf{L}$  is the standard 7-pt discrete Laplace operator,  $\mathbf{n} \in \mathbb{Z}^3$ , and  $\Omega_h$  is a bounded domain in  $\mathbb{Z}^3$ . In practice, the constraint on  $\text{supp}(f)$  can be relaxed by prescribing a finite tolerance and requiring that all non-negligible sources, i.e. sources with a magnitude greater than the prescribed tolerance, be located in a bounded region. The solution to Eq. 4 is given by

$$u(\mathbf{n}) = [\mathbf{G} * f](\mathbf{n}) = \sum_{\mathbf{m} \in \Omega_h} \mathbf{G}(\mathbf{n} - \mathbf{m}) f(\mathbf{m}), \quad (5)$$

where  $\mathbf{G}(\mathbf{n})$  is the fundamental solution of the discrete Laplace operator.

An expression for  $\mathbf{G}(\mathbf{n})$  in terms of Fourier integrals is provided by

$$\mathbf{G}(\mathbf{n}) = \frac{1}{8\pi^3} \int_{[-\pi, \pi]^3} \frac{\exp(-i\mathbf{n} \cdot \boldsymbol{\xi})}{2 \cos(\xi_1) + 2 \cos(\xi_2) + 2 \cos(\xi_3) - 6} d\boldsymbol{\xi}. \quad (6)$$

The expression in Eq. 6 is readily obtained by using Discrete Fourier Transforms (DFTs) to diagonalize and invert  $\mathbf{L}$ , and applying appropriate limiting procedures to convert infinite sums into integrals. Details regarding the construction of Eq. 6 and expressions for the fundamental solutions to other discrete operators are found in [10, 19, 8]. Additionally, Appendix A provides an outline of the numerical procedures used to evaluate  $\mathbf{G}(\mathbf{n})$  for small values of  $|\mathbf{n}|$ .

Although it is possible to compute  $\mathbf{G}(\mathbf{n})$  by numerically evaluating Eq. 6, for large values of  $|\mathbf{n}|$  it is more efficient to evaluate the LGF via its asymptotic expansion. Techniques for constructing asymptotic expansions of LGFs to arbitrary order are described in [19, 20, 8]. The asymptotic expansion of  $\mathbf{G}(\mathbf{n})$  in 3D is given by

$$A_G(\mathbf{x}) = -\frac{1}{4\pi|\mathbf{x}|} - \frac{x_1^4 + x_2^4 + x_3^4 - 3x_1^2x_2^2 - 3x_1^2x_3^2 - 3x_2^2x_3^2}{16\pi|\mathbf{x}|^7} + \mathcal{O}(|\mathbf{x}|^{-5}), \quad (7)$$

where  $\mathbf{x} = (x_1, x_2, x_3)$ . In subsequent discussions,  $A_G^n(\mathbf{x})$  denotes the asymptotic expansion of  $\mathbf{G}(\mathbf{n})$  corresponding to the first  $n$ -terms of  $A_G(\mathbf{x})$ . For a fixed value number of terms,  $n$ , it can be shown that  $|A_G^n(\mathbf{n}) - \mathbf{G}(\mathbf{n})| \sim \mathcal{O}(|\mathbf{n}|^{-2n-1})$  [8]. As expected, the first term in Eq. 7 corresponds the fundamental solution of the Laplace operator.

Despite the fact that  $G(\mathbf{x})$  and  $\mathbf{G}(\mathbf{n})$  share the same asymptotic behavior, there are significant differences in their behavior near the origin. Unlike  $G(\mathbf{x})$ , which is singular at the origin,  $\mathbf{G}(\mathbf{n})$  remains finite for all values of  $\mathbf{n}$ .  $G(\mathbf{x})$  is scale-invariant, i.e. there exists a  $k$  such that  $G(\alpha\mathbf{x}) = \alpha^k G(\mathbf{x})$ , whereas  $\mathbf{G}(\mathbf{n})$  is not scale-invariant. Furthermore,  $G(\mathbf{x})$  is spherical symmetric about the origin, as opposed to  $\mathbf{G}(\mathbf{n})$  which has reflectional symmetry about the principal axes and is invariant under index permutations.

In addition to providing expressions for the fast evaluation of LGFs, asymptotic expansions of LGFs allow for the sum given in Eq. 5 to be decomposed into three parts,

$$\mathbf{u}(\mathbf{n}) = \mathbf{u}^{\text{direct}}(\mathbf{n}) + u^{\text{asympt},n}(\mathbf{n}) + \epsilon(\mathbf{n}), \quad (8)$$

where

$$\mathbf{u}^{\text{direct}}(\mathbf{n}) = \sum_{\mathbf{m} \in \Omega_h^{\text{direct}}(\mathbf{n})} \mathbf{G}(\mathbf{n} - \mathbf{m})\mathbf{f}(\mathbf{m}), \quad (9)$$

$$u^{\text{asympt},n}(\mathbf{n}) = \sum_{\mathbf{m} \in \Omega_h \setminus \Omega_h^{\text{direct}}(\mathbf{n})} A_G^n(\mathbf{n} - \mathbf{m})\mathbf{f}(\mathbf{m}), \quad (10)$$

and  $\epsilon(\mathbf{n})$  is the error due to approximating  $\mathbf{G}(\mathbf{n})$  with  $A_G^n(\mathbf{n})$  over the region  $\Omega_h \setminus \Omega_h^{\text{direct}}$ . The region  $\Omega_h^{\text{direct}}(\mathbf{n})$  is a subset of  $\Omega_h$  for which the LGF is evaluated directly, i.e. via numerical evaluation of Eq. 6, as opposed to being evaluated via its asymptotic expansion. Typically, the region  $\Omega_h^{\text{direct}}(\mathbf{n})$  is defined by a small cubic box centered at the grid point  $\mathbf{n}^1$ . The first term of Eq. 8,  $\mathbf{u}^{\text{direct}}(\mathbf{n})$ , is a grid function evaluated at the grid point  $\mathbf{n}$ , whereas the second term,  $u^{\text{asympt},n}(\mathbf{n})$ , is a continuous function evaluate at the

---

<sup>1</sup>For the case of the 3D discrete Laplace operator, choosing  $\Omega_h^{\text{direct}}$  to be a cubic box with side lengths of 14, 41, and 134 grid points is sufficient to achieve relative errors less than  $10^{-5}$ ,  $10^{-10}$ , and  $10^{-15}$ , respectively, using the five term asymptotic expansion. The size of  $\Omega_h^{\text{direct}}$  can be reduced by including more terms in the asymptotic expansion, for example, a box with side lengths of 38 grid points and the thirteen term asymptotic expansion achieve relative error less than  $10^{-15}$ .

location of the grid point  $\mathbf{n}$ . As will be discussed in subsequent sections, this decomposition allows for  $u^{\text{asympt},n}(\mathbf{n})$  to be evaluated using fast techniques developed for continuous kernels.

## 2.2. Fast convolutions on regular grids via FFTs

Consider the one-dimensional the convolution given by

$$u(x_i) = \sum_{j=0}^{M-1} K(x_i, y_j) f(y_j), i = 0, 1, \dots, N-1. \quad (11)$$

where  $x_i = x_0 + ih$  for  $i = 0, 1, \dots, N-1$ , and  $y_j = y_0 + jh$  for  $j = 0, 1, \dots, M-1$ . If the kernel  $K(x, y)$  is translation invariant, i.e.  $K(x, y) = K(x - y)$ , then Eq. 11 can be expressed as the discrete convolution between two vectors,

$$u_i = \sum_{j=0}^{M-1} k_{N-1+j-i} f_j, i = 0, 1, \dots, N-1, \quad (12)$$

where  $u_i = u(x_i)$ ,  $f_j = f(y_j)$ , and  $k_{N-1+j-i} = K(x_j - y_i)$ . Discrete linear convolutions of this form can be casted into circular convolutions by appropriate padding of the vectors  $u$  and  $f$ . The discrete version of the Convolution Theorem can then be used to compute the DFT of the circular convolution as the product of two vectors. These observations, and the ability to perform DFTs via FFTs, leads to a fast FFT-based convolution technique given by

1. Pad sequence with zeros: append  $N-1$  zeros to vector  $f$ .

$$\bar{f}_i = [\text{Pad}(f)]_i = \begin{cases} f_i & i = 0, 1, \dots, N-1 \\ 0 & i = N, N+1, \dots, N+M-2 \end{cases} \quad (13)$$

2. Forward DFT: compute the DFT of sequences  $\bar{f}$  and  $k$  via FFTs.

$$\hat{f} = \text{FFT}(\bar{f}), \hat{k} = \text{FFT}(k) \quad (14)$$

3. Convolution of DFTs: multiply complex coefficients of  $\hat{f}$  and  $\hat{k}$ .

$$\hat{u}_i = [\text{Prod}(g, k)]_i = \hat{f}_i \hat{k}_i, i = 0, 1, \dots, N+M-2 \quad (15)$$

4. Backward DFT: compute the inverse DFT of sequence  $\hat{u}$  via FFT.

$$\bar{u} = \text{FFT}^{-1}(\hat{u}) \quad (16)$$

5. Truncate sequence: remove the first  $M - 1$  entries of  $\bar{u}$  to obtain  $u$ .

$$u_i = [\text{Trunc}(\bar{u})]_i = \bar{u}_{M+i}, i = 0, 1, \dots, N - 1 \quad (17)$$

This technique is considered to be fast since it requires  $\mathcal{O}((N + M) \log(N + M))$  operations. Furthermore, each step of the procedure is readily generalized to higher dimensions for the case of tensor-product grids by recursively applying the 1D version to each directions.

### 2.3. Adaptive block-structured grid

Fast convolutions via FFTs discussed in the previous section can be used to accelerate the evaluation of Eq. 5. In order to use this technique, the support of  $\mathbf{f}$  needs to be padded with zeros to form a box. Similarly, the region where  $\mathbf{u}$  is evaluated needs to be extended to also form a box. For cases where the domain defined by the support of the source terms is not a box, the cost of the additional computational elements can outweigh the reduced operation count per grid point of the FFT-based convolution technique.

Computational domains defined by the union of blocks can, however, be used to avoid excessive padding and still retain sufficient regularity to benefit from the fast FFT-based convolution technique. Our formulation partitions the infinite grid into blocks defined on a logically Cartesian grid. Blocks can potentially have a different number of grid points in each direction, but all blocks are required have the same dimensions. An active source block denotes a block containing non-zero sources. Similarly, an active evaluation block denotes a block containing grid points on which the induced field is evaluated. The union of active source (evaluation) blocks is referred to as the active source (evaluation) grid.

Let  $B_s$  and  $B_e$  denote the sets of active source and evaluation blocks, respectively. The convolution given in Eq. 5 can be evaluated by

$$u^P = \sum_{Q \in B_s} \text{conv}(k^{Q-P}, f^Q), \quad \forall P \in B_e, \quad (18)$$

where  $u^P$  and  $f^P$  denote vectors containing the values of  $\mathbf{u}(\mathbf{n})$  and  $\mathbf{f}(\mathbf{n})$ , respectively, evaluated on the grid points belonging to block  $P$ . Similarly,  $k^{Q-P}$  denotes the vector containing the unique values of  $\mathbf{G}(\mathbf{m} - \mathbf{n})$ , as described in previous section, for values of  $\mathbf{n}$  and  $\mathbf{m}$  corresponding to the indices of grid points belonging to block  $Q$  and  $P$ , respectively. If blocks  $P$  and  $Q$  are sufficiently well-separated then  $A_G^n(\mathbf{m} - \mathbf{n})$  is used instead of  $\mathbf{G}(\mathbf{m} - \mathbf{n})$



for constructing  $k^{Q-P}$ . The operator  $\text{conv}(k^{Q-P}, f^Q)$  denotes the generalization of Eq. 11 to arbitrary dimensions. Details regarding the construction of vectors  $u^P$ ,  $f^P$ , and  $k^{Q-P}$  are omitted, since it immediately follows the discussion regarding the tensor-product grid generalization of Eq. 11.

Computing each instance of  $\text{conv}(k^{Q-P}, f^Q)$  in Eq. 18 using the fast FFT-based convolution technique leads to a scheme that evaluates Eq. 5, for the case of  $B_s = B_e$ , in  $\mathcal{O}(N_B^2 N_b \log(N_b))$  operations, where  $N_B$  is the number of blocks belonging to  $B_s$ , and  $N_b$  is the number of grid points belonging to each block. The operation count can be further reduced to  $\mathcal{O}(N_B N_b \log(N_b) + N_B^2 N_b)$  if the DFT of the kernel blocks,  $\hat{k}^{Q-P}$ , are pre-computed, and if the DFT of source and evaluation blocks are preformed as pre-processing and post-processing step, respectively. Details regarding pre-computations, and pre- and post-processing steps are discussed in subsequent sections.

#### 2.4. Fast convolutions via interpolation-based kernel compression

Interpolation-based FMMS obtain a low-rank representation of the kernel,  $K(\mathbf{x}, \mathbf{y})$ , by projecting it onto a finite basis of interpolation functions. Consider a function  $f(\mathbf{x})$  sampled at  $n$  points,  $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{n-1}$ . An approximation for  $f(\mathbf{x})$  is given by

$$\tilde{f}^n(\mathbf{x}) = \sum_{i=0}^{n-1} \phi_i(\mathbf{x}) f(\mathbf{x}_i), \quad (19)$$

where  $\phi_i(\mathbf{x})$  is a interpolation function associated with the interpolation node  $\mathbf{x}_i$ . An approximation for  $K(\mathbf{x}, \mathbf{y})$  is obtained by recursively applying Eq. 19 to each argument of  $K(\mathbf{x}, \mathbf{y})$ ,

$$\tilde{K}^n(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \psi_i(\mathbf{x}) K(\mathbf{x}_i, \mathbf{y}_j) \phi_j(\mathbf{y}), \quad (20)$$

where  $\{\phi\} = \{\phi_0, \phi_1, \dots, \phi_{n-1}\}$  and  $\{\psi\} = \{\psi_0, \psi_1, \dots, \psi_{n-1}\}$  are potentially distinct bases of interpolation functions. In order to make the kernel-compression technique symmetric, only schemes with  $\{\phi\} = \{\psi\}$  are considered. Directly applying this kernel compression technique to discrete convolutions of the form of Eq. 1, leads to an approximation of  $u(\mathbf{x}_i)$  given by

$$u(\mathbf{x}_i) \approx \sum_{j=0}^{M-1} \sum_{p=0}^{n-1} \sum_{q=0}^{m-1} \phi_p(\mathbf{x}_i) K(\mathbf{x}_p, \mathbf{y}_q) \phi_q(\mathbf{y}_j) f(\mathbf{y}_j), \quad i = 0, 1, \dots, N-1. \quad (21)$$

For cases involving multiple sets of either evaluation points,  $\mathbf{x}_i$ , or source points,  $\mathbf{y}_j$ , it is advantageous to decompose the evaluation of Eq. 21 into three steps:

1. Regularization: compute effective source terms using the adjoint of the interpolation procedure.

$$\tilde{f}(\mathbf{y}_q) = \sum_{j=0}^{M-1} \phi_q(\mathbf{y}_j) f(\mathbf{y}_j), \quad q = 0, 1, \dots, n-1 \quad (22)$$

2. Convolution: compute the field induced by effective source terms on interpolation nodes.

$$\tilde{u}(\mathbf{x}_p) = \sum_{q=0}^{n-1} K(\mathbf{x}_p, \mathbf{y}_q) \tilde{f}(\mathbf{y}_q), \quad p = 0, 1, \dots, n-1 \quad (23)$$

3. Interpolation: compute the field at evaluation points using the interpolation procedure.

$$u(\mathbf{y}_i) = \sum_{p=0}^{n-1} \phi_p(\mathbf{x}_i) \tilde{u}(\mathbf{x}_p), \quad i = 0, 1, \dots, N-1 \quad (24)$$

If the values of  $\phi_q(\mathbf{x}_i)$  and  $\phi_q(\mathbf{y}_j)$  are known, the number of operations required by this procedure, for the case of  $M = N$ , is  $\mathcal{O}(nN + n^2)$ . For the case of  $n \ll N$  this procedure represents a significant reduction in the number of operations compared to straightforward method of evaluating Eq. 1.

### 2.5. Fast convolution on regular grids using polynomial interpolation and FFTs

The fast convolution techniques presented in Sections 2.2 and 2.4 can be combined to yield a faster method for evaluating the block-wise convolutions involved in Eq. 18. This technique follows from the observation that Eq. 23 can be evaluated using FFTs if the kernel is translation-invariant and the interpolation nodes on which the kernel is evaluated are restricted to be on a regular grid. The first requirement is assumed since the kernels corresponding to the fundamental solutions of the linear, constant-coefficient elliptic difference equations are translation-invariant. The second condition is achieved by using an interpolation scheme based on equidistant interpolation nodes

on tensor-product grids. Although many interpolation schemes satisfy the latter condition, only polynomial interpolation schemes on tensor-product grids are presently considered since they are fast, simple to implement, and their behavior is well-understood<sup>2</sup>.

Polynomial interpolation on tensor-product grids is performed by recursively applying 1D polynomial interpolation along each direction. This generalization has the advantage of maintaining the number of operation per grid point independent of dimension, and allows the behavior of the interpolation process to readily generalized from its 1D version.

In the absence of rounding errors, 1D polynomial interpolants converge geometrically if the function being interpolated is analytic in a region on the complex plane near the interpolation interval. The size and shape of this convergence region depends on the choice of interpolation nodes [26]. The kernels being considered correspond to the asymptotic expansion of the LGFs that are only discontinuous at the origin. Although convergences conditions should be verified for each kernel, the requirement that source and evaluation blocks be sufficiently well-separated to accurately evaluate the LGF using its asymptotic expansion is often sufficient to guarantee convergence.

Unlike Chebyshev interpolation, polynomial interpolation on equidistant nodes is ill-conditioned. Ill-conditioning can cause rounding errors due finite numeric precision to be amplified. The Lebesgue constant,  $\Lambda_n(X)$ , for a set of  $n$  interpolation points  $X = x_0, x_1, \dots, x_{n-1}$ , can be used to bound the growth of perturbations in the data [27],

$$\max_{x \in I} |p(x) - \tilde{p}(x)| \leq \Lambda_n(X) \max_{0 \leq i \leq n-1} |f(x_i) - \tilde{f}(x_i)|, \quad (25)$$

where  $I$  is the interpolation interval,  $p(x)$  and  $\tilde{p}(x)$  are the polynomials interpolants resulting from the nodal values  $f(x_i)$  and  $\tilde{f}(x_i)$ , respectively. The Lebesgue constant is given by

$$\Lambda(X) = \max_{x \in I} \sum_{i=0}^{n-1} |\phi_i(x)|, \quad (26)$$

---

<sup>2</sup>We have not explored alternative interpolation procedures, with the exception of Fourier interpolation on non-periodic domains. Although Fourier interpolation would be particularly appropriate given FFTs are used in our method to accelerate local computations, preliminary result have shown that this procedure is less efficient (in terms of points per unit accuracy) than the procedure described in this section.

where  $\phi_i(x)$  is the Lagrange characteristic polynomial associate with  $x_i$ . Equation 25 can be extended to polynomial interpolation on tensor product grids, with equal number of points and spacing in each direction, by replacing  $\Lambda(X)$  with  $(\Lambda_n(X))^d$ , where  $d$  is the dimension of the problem.

In the limit of very large  $n$ , the Lebesgue constant of a set of equally spaced nodes is known to grow exponentially [27]. In order to avoid very large Lebesgue constants, the present scheme restricts number nodes used for polynomial interpolation to be at most  $n_{max}$ . If  $n_{max}$  nodes are insufficient to achieve a desired interpolation error, additional nodes are added to the interval, but only the closest  $n_{max}$  nodes to the evaluation point are used for interpolation. Thus, this hybrid scheme performs both  $p$ - and  $h$ -refinement to increase the accuracy of the interpolations procedure. As a result, geometric convergence rates are expected for  $n \leq n_{max}$ , and polynomial convergence rates of order  $n_{max} - 1$  are expected for  $n > n_{max}$ . The values of  $n$  and  $n_{max}$  required to interpolate a function  $f(x)$  over an interval  $I$  with an interpolation error less than  $\epsilon$  are obtained in two steps:

1. Find the largest  $n_{max}$  such that  $\Lambda_{n_{max}}(X)\epsilon_p$  is less than  $\epsilon$ , where  $\epsilon_p$  is the precision of the floating-point scheme.
2. Progressively increase the number of  $n$  until the difference between  $f(x)$  and it approximation are less than  $\epsilon$ .

The same procedure can be used in higher dimensions by having  $n$  and  $n_{max}$  correspond to the number of interpolation points along each direction, and replacing  $\Lambda_{n_{max}}(X)$  with  $(\Lambda_{n_{max}}(X))^d$ . For example, approximating a function in 3D using double-precision arithmetic to relative tolerance of  $\epsilon = 1.25 \times 10^{-12}$  requires  $n_{max} \leq 10$ .

We omit a step-wise description of the combined fast algorithm for block-wise convolutions, since it readily follows from the discussion. Instead, we introduce the notation  $\tilde{f}^P = \text{Interp}(f^Q)$  and  $f^Q = \text{Reg}(\tilde{f}^P)$  corresponding to the interpolation and regularization (adjoint of interpolation) operations, respectively. Vector  $f^P$  contains the values of  $\mathbf{f}(\mathbf{n})$  evaluated on the grid points of block  $P$ . Similarly, vector  $\tilde{f}^Q$  contains the values of  $\tilde{f}(\mathbf{x})$  evaluated on the grid points of the interpolation interval  $Q$ .

### 3. The Fast Lattice Green's Function method

#### 3.1. Basic algorithm

Thus far we have discussed methods for accelerating the evaluation Eq. 18 by performing fast block-wise convolutions involving interpolation-based ker-

nel compression and/or FFT techniques. Asymptotically these schemes require  $\mathcal{O}(N^2)$  operations, though the constant in front of the  $N^2$  term can be significantly smaller compared to that of the straightforward method. For kernels that decay or exhibit progressively smoother behavior away from the origin, for example the fundamental solution of the discrete Laplace operator, it is possible to combine the fast block-wise convolution techniques of the previous sections with the multilevel scheme of the original FMM [21]. To facilitate the discussion, we will assume that the active evaluation grid is the same as the active source grid.

Our multilevel scheme follows a tree (octree in 3D) structure similar to that described in [21]. Tree nodes at all levels are said to correspond to intervals<sup>3</sup> Each interval is defined by the tensor-product grid associated with the nodes of the interpolation scheme. The tree is constructed by first creating one tree leaf, i.e. tree node with no children, for each active grid block. The intervals of tree leaves are defined such that they occupy the same spatial region as their associated grid blocks. After defining all tree leaves, sibling are recursively merged to generate the multilevel structure. We use the convention that all tree leaves are located at level 1 and that the root of the tree is located at level  $L$ .

The set of intervals at level  $\ell$  is denoted by  $B^\ell$ , and  $N_B^\ell$  denotes the size of  $B^\ell$ . In order to facilitate the discussion, intervals at level  $\ell$  are chosen to contain  $n_b^\ell$  nodes in each directions. The total number of points in each interval at level  $\ell$  is given by  $N_b^\ell = (n_b^\ell)^d$ , where  $d$  is the dimension of the problem. The set of blocks defining the underlying active grid is denoted by  $B^0$ .  $N_B^0$ ,  $n_b^0$ , and  $N_b^0$  have analogous definitions to  $N_B^\ell$ ,  $n_b^\ell$ , and  $N_b^\ell$ , respectively. We note that level zero,  $\ell = 0$ , is not part of the tree structure, but the slight abuse of notation facilitates the description of the algorithm.

By construction, all intervals are Cartesian grids. As a result the union of intervals belonging to the same level defines an analogous grid to that of the underlying adaptive block-structured grid. Therefore, the techniques for fast block-wise convolutions discussed in previous sections are readily generalized to all levels of the tree structure. The overall algorithm for solving systems of difference equations on adaptive block-structured grids, referred to as the

---

<sup>3</sup>In the context of the hierarchical algorithm and structure, the term “interval” is equivalent to term “box” used in [21]. We reserve the term “box” for geometric descriptions, and do not associate any specific structure or information with the term.

Fast Lattice Green's Function (FLGF) method, is given by

0. *Pre-computation*: compute and store all unique  $\hat{k}^{P-Q}$  used in Step 2.

$$\hat{k}^P = \text{FFT}(k^{P-Q}) \quad (27)$$

1. *Upwards Pass*:  $\forall P \in B^\ell$ , for  $\ell = 0, 1, \dots, L$ 
  - (a) Regularize: compute effective source terms at interpolation nodes

$$\tilde{f}^P = \sum_{Q \in \text{RegSupp}(P)} \text{Reg}(\tilde{f}^Q), \quad (28)$$

- (b) Padded forward DFT: prepare vectors for DFT convolutions

$$\hat{f}^P = \text{FFT}(\text{Pad}(\tilde{f}^P)), \quad (29)$$

2. *Level Interactions*:  $\forall P \in B^\ell$ , for  $\ell = 0, 1, \dots, L$

$$\hat{u}^P = \sum_{Q \in \text{InfluenceList}(P)} \text{Prod}(\hat{f}^P, \hat{k}^{P-Q}), \quad (30)$$

3. *Downwards Pass*:  $\forall P \in B^\ell$ , for  $\ell = L, L-1, \dots, 0$

- (a) Truncated backwards DFT: extract relevant data from DFT convolution

$$\tilde{v}^P = \text{Trunc}(\text{FFT}^{-1}(\hat{u}^P)) \quad (31)$$

- (b) Interpolate: compute and aggregate the induced field at interpolation nodes

$$\tilde{u}^P = \tilde{v}^P + \text{Interp}(\tilde{u}^{\text{InterpSupp}(P)}) \quad (32)$$

We note that the operations performed in steps 1, 2, and 3 are commonly referred to as the multipole-to-multipole, multipole-to-local, and local-to-local operations, respectively, in the FMM literature. The lists of blocks/intervals used by the algorithm are given by

$$\text{RegSupp}(P) = \begin{cases} \text{block } P & \text{if } P \in B^0 \\ \text{block associated with interval } P & \text{if } P \in B^1 \\ \text{children of interval } P & \text{if } P \in B^\ell, 2 \leq \ell \leq L \end{cases} \quad (33)$$

$$\text{InterpSupp}(P) = \begin{cases} \text{interval associated with block } P & \text{if } P \in B^0 \\ \text{parent of interval } P & \text{if } P \in B^\ell, 1 \leq \ell < L \\ \emptyset & \text{if } P \in B^L \end{cases} \quad (34)$$

$$\text{InfluenceList}(P) = \begin{cases} \text{near-neighbors of block } P & \text{if } P \in B^0 \\ \text{interaction list of interval} & \text{if } P \in B^\ell, 1 \leq \ell \leq L \end{cases} \quad (35)$$

Children, parents, near-neighbors, and interaction lists follow the same definitions those of the original FMM [21]. Based on these definitions, we note that Eq. 28 reduces to  $\tilde{f}^P = f^P$  for  $P \in B^0$ , and Eq. 32 reduces to  $\tilde{u}^P = \tilde{v}^P$  for  $P \in B^L$ .

### 3.2. Algorithmic complexity

The overall complexity of our algorithm is  $\mathcal{O}(N)$ , as is the case for the original FMM. For simplicity, the discussion concerning the cost of each step is limited to the 3D version of the algorithm. Details regarding to the cost of each block/interval are presented in Table 1. The factor of 8 in front of  $C_{\text{Interp}}^\ell$  for the *Upwards Pass* ( $\ell > 1$ ) is due to the fact that each interval has 8 children. The constants, 27 and 189, associated with the cost of *Level Interactions* correspond to the number of near-neighbors and the number of members of each interaction list, respectively.

Table 1: Cost per interval/block of present FMM method.

	cost	order
Pre-computations	$C_{\text{EvalKernel}}^\ell + C_{\text{PadFFT}}^\ell$	$N_b^\ell \log N_b^\ell$
Upwards pass ( $\ell = 0$ )	$C_{\text{PadFFT}}^0$	$N_b^0 \log N_b^0$
Upwards pass ( $\ell = 1$ )	$C_{\text{Interp}}^\ell + C_{\text{PadFFT}}^\ell$	$N_b^\ell \log N_b^\ell$
Upwards pass ( $\ell > 1$ )	$8C_{\text{Interp}}^\ell + C_{\text{PadFFT}}^\ell$	$N_b^\ell \log N_b^\ell$
Level interactions ( $\ell = 0$ )	$27C_{\text{Prod}}^\ell$	$27N_b^0$
Level interactions ( $\ell > 0$ )	$189C_{\text{Prod}}^\ell$	$189N_b^\ell$
Downwards pass	$C_{\text{Interp}}^\ell + C_{\text{PadFFT}}^\ell$	$N_b^\ell \log N_b^\ell$

The specific values and a brief discussion of the constants presented in Table 1 is provided:

1.  $C_{\text{EvalKernel}}^\ell$ : Cost of kernel evaluation performed in Eq. 27. Constructing the vector  $k^{Q-P}$ , where  $Q$  and  $P$  are blocks/intervals at level  $\ell$ , requires  $8N_b^\ell$  kernel evaluations. For small values of  $|\mathbf{n} - \mathbf{m}|$  a look-up table is used to evaluate  $\mathbf{G}(\mathbf{n} - \mathbf{m})$ , otherwise there kernel is evaluated using  $A_G^n(\mathbf{n} - \mathbf{m})$ .

2.  $C_{\text{Interp}}^\ell$ : Cost of polynomial interpolation performed in Eq. 32. The coefficient mapping interpolation nodes to evaluation nodes are pre-computed (only needed for 1D interpolation); therefore computing the values of a single block/interval at level  $\ell$  requires  $\min(n_b^{\ell+1}, n_{\text{max}})N_b^\ell$  operations (1 real addition and 1 real multiplication per operation), where  $n_{\text{max}}$  is described in Section 2.5. In 3D,  $n_{\text{max}}$  is typically set to be no greater than 10.  $C_{\text{Interp}}^\ell$  also describes the cost of performing the regularization, adjoint of interpolation, operation involved in each term of the sum in Eq. 28.
3.  $C_{\text{PadFFT}}^\ell$ : Cost of performing a 3D FFT (real-to-complex) or inverse FFT (complex-to-real) on the padded vectors present in Eq. 27, 29, and 31. The operation count (total number of real additions and multiplications) for each FFT performed using the FFTW library is approximately  $2(8N_b^\ell) \log_2(8N_b^\ell)$  [28], where  $8N_b^\ell$  is the size of the padded vectors. Since all FFTs are real-to-complex or complex-to-real approximately half of the coefficients are redundant and need neither be stored nor operated on.
4.  $C_{\text{Prod}}^\ell$ : Cost of performing DFT convolutions, i.e. multiplication of complex coefficients, in Eq. 30. If blocks/intervals  $Q$  and  $P$  belong to level  $\ell$ , then performing  $\text{Prod}(\hat{f}^P, \hat{k}^{P-Q})$  requires approximately  $4N_b^\ell$  operations (1 complex multiplication per operation), where  $4N_b^\ell$  is the number of non-redundant DFT coefficients per block/interval.

The cost per level of each operation, except for the *Pre-computation* step, can be obtained by multiplying the values of each row by the  $N_B^\ell$ . In regards to the *Pre-computation* step, the cost per level for  $\ell = 0$  and  $\ell > 0$  is obtained by multiplying the cost per block/interval by 27 and 317, respectively, which correspond to the number of unique  $\hat{k}^{P-Q}$  vectors that are used at each level. If the kernel shares the same symmetry as the LGF of the discrete Laplace operator, the number of unique  $\hat{k}^{P-Q}$  vectors per level is reduced to 4 and 36 for  $\ell = 0$  and  $\ell > 0$ , respectively. If symmetry is used to reduce number of pre-computed  $\hat{k}^{P-Q}$  vectors, then  $C_{\text{Prod}}^\ell$  is roughly doubled since a twiddle factor needs to be applied to the DFT coefficients for cases involving reflections.

### 3.3. Parallel implementation

It is not in the scope of this paper to provide a detail description of the parallel implementation of the FLGF method. However, a brief overview



of our MPI-based algorithm is included to demonstrate that our method allows for a simple parallel implementation suitable for practical large-scale scientific computing.

Our current implementation builds the tree structure and performs the load balancing operations on a single MPI-process, and then scatters the information to all other process. As a result, all the information necessary to evaluate RegSupp, InterpSupp, and InfluenceSupp for any block/interval is known by all processes. The tree structure is constructed following the bottom-up approach discussed in Section 3.1. Prior to partitioning the problem, the load balancing scheme first assigns a weight to every block and interval based on an estimate of its runtime cost. Next, parent tree nodes are recursively grouped with one of its child tree nodes, and tree leaves are grouped with their associated grid blocks. The set of groups is then partitioned into clusters in such away that weight of each cluster (aggregate weight of all interval/blocks belonging to all groups in the cluster) is roughly the same. Each cluster is then assigned to an MPI process. Given each group contains only one block, a Morton or Z-order curve [29] is used to preserve data locality during partitioning.

The parallel algorithm closely resembles the serial algorithm, since each process executes steps analogous to the *Upwards Pass*, *Level Interactions*, and *Downwards Pass*. Non-blocking routines are used for all communications, allowing for computations to be overlapped with communications. Furthermore, our algorithms uses these routines to avoid any global synchronization within each steps and between steps.

The parallel execution of each step follows a similar event-driven paradigm, where processes performs “work units” based on the information they currently have and send information to other processes, or to itself, when sufficient “work units” have been completed. Send and receive buffers are used to avoid excessive memory requirements. Our algorithm gives priority to “work units” yielding results that are sent to other processes. The time spent waiting to either receive information or to clear send buffers is used to perform local “work units”, i.e. operations that only require data and yield results pertinent to the same process. During the *Upward* and *Downward Pass* a non-blocking send is posted after each interval-/block-wise regulation and interpolation operation is completed. In contrast, during *Level Interactions* step the influence of all intervals/blocks belonging to a process on all intervals/blocks belonging another process is aggregated and packages before sending; thus each process sends at most one message to every other process

during this step.

#### 4. Numerical results

In this section we present numerical results that demonstrate the accuracy, computational cost, and parallel performance of the FLGF method. We solve the discrete Poisson equation in 3D. As in the previous section, the active evaluation grid is defined to be equal to the active source grid. For all test cases, blocks/intervals belonging to level  $\ell$  are chosen to contain  $n_b^\ell$  points in each directions. In the interest of brevity, we only consider schemes where the number of interpolation nodes per interval is same for all levels,  $n_b^\ell = n_I$  for  $\ell = 1, 2, \dots, L$ , and require that  $n_I = n_b^0 + 1^4$ . These considerations reduce the parameter space of possible schemes to cases where spacing between nodes of parent intervals is twice that of child intervals. Furthermore, there is effectively no regularization/interpolation between level 0 and level 1, since the nodes on level 1 coincide with the underlying grid points. Following the discussion of Section 2.5 we set maximum number nodes used for interpolating a value at single point,  $n_{\max}$ , to be 10 for all test cases.

We note that the previous restrictions are only imposed for the purpose of a concise exposition. The non-scale-invariant behavior of most LGFs suggests that, in general, non-trivial performance gains can be achieved by tuning  $n_b^\ell$  for each level. Given the LGF of the discrete Laplace operator is approximately scale-invariant away from the origin, possible performance gains achieved by varying  $n_b^\ell$  are not considered in the present discussion.

Our code is written in Fortran, and uses the third-party libraries MVA-PICH2 and FFTW 3.3 to perform MPI-based commutations and to evaluate DFTs via FFTs, respectively. Numerical experiments are performed using our local computing facility which consists of 60 compute nodes connected by a QDR InfiniBand network. Each node containing 2 Intel Xeon X5650 processor (6-core, 12MB cache, 2.66GHz clock speed) and 48GB of RAM.

##### 4.1. Error

The accuracy of the proposed methodology is investigated on cubic active grids containing different numbers of active grid points and partitioned

---

<sup>4</sup>Our implementation requires for intervals belonging to  $\ell > 0$  have a one grid point overlap with its neighboring intervals along  $(n_1, n_2, n_3)$  directions, where  $n_i = \{0, 1\}$  and at least one  $n_i$  is non-zero.

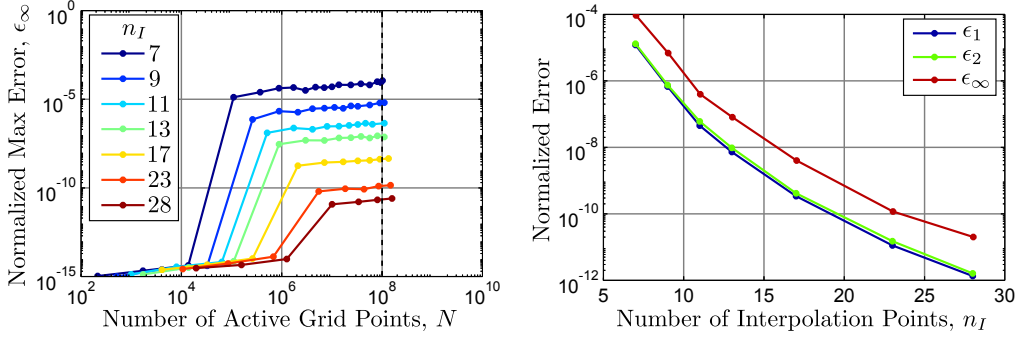


Figure 1: *Left*: max error,  $\epsilon_\infty$ , for cubic active grids containing  $N$  grid points partitioned into blocks with  $n_b^0 = n_I - 1$  grid points along each direction. *Right*: error for test cases containing  $10^8$  grid points as function of  $n_I$ ; the curve corresponding to  $\epsilon_\infty$  is equivalent to the values on the *left* plot intersecting the vertical dashed line.

into blocks of different sizes. A procedure based on random manufactured solutions is used to determine the error of each test case. In this procedure a solution,  $\mathbf{u}_{\text{rand}}(\mathbf{n})$ , is manufactured by assigning a random value between  $-1$  and  $+1$  to each grid points, except for grid points on the boundary of the active grid which are set to zero. The source distribution,  $\mathbf{f}(\mathbf{n})$ , which serves as the the input for each test case, is computed by taking the discrete Laplacian of the prescribed solution, i.e.  $\mathbf{f}(\mathbf{n}) = [\mathbf{L}\mathbf{u}_{\text{rand}}](\mathbf{n})$ . The error for each test case is computed by  $\epsilon_p = \|\mathbf{u} - \mathbf{u}_{\text{rand}}\|_p / \|\mathbf{u}_p\|_p$ , where  $\mathbf{u}(\mathbf{n})$  is determined by solving  $[\mathbf{L}\mathbf{u}_p](\mathbf{n}) = \mathbf{f}(\mathbf{n})$ , and  $\|\mathbf{u}\|_p$  is the  $L^p$ -norm of  $\mathbf{u}$  computed over the active grid. The error for various problem sizes and schemes based on different blocks sizes is presented in Fig. 1.

Test cases involving a small number blocks do not make use of the interpolation-based kernel compression technique; they only make use of the block-wise FFT-based convolution technique, which incurs an error close to machine precision. As a result, the three cases with the smallest values of  $N$  for each series have significantly smaller errors compared to their respective asymptotic (large  $N$ ) error.

Given that we chose  $n_{\text{max}} = 10$  and constrained  $n_b^0$  to be proportional  $n_I$ , the error is expected to decay as  $n_I^{-10}$  for  $n_I > n_{\text{max}}$  (one order greater than interpolation order due to the  $\sim |\mathbf{x}|^{-1}$  decay of the LGF). The data shown Fig. 1 are consistent with our estimates, exhibiting a behavior proportional to  $n_I^{-10.7}$  for values of  $n_I$  between 11 and 28. The significant difference in the magnitude of  $\epsilon_1$  and  $\epsilon_2$  compared to  $\epsilon_\infty$  suggests that the maximum error is

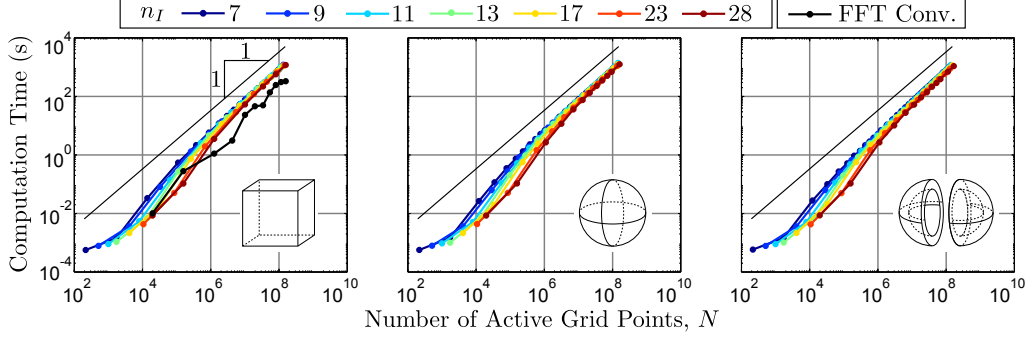


Figure 2: Computation times for active grids containing  $N$  grid points partitioned into blocks with  $n_b^0 = n_I - 1$  point in each direction. The curve labeled “FFT Conv.” corresponds to the special cases where the entire active grid is a single block. Results are presented for active grids with geometries approximating: cubes (*left*), spheres (*middle*), and spherical-shells (*right*).

concentrated in lower dimensional regions of the grid. Spatial plots of the error (not included) confirm that larger errors are always observed near or on the boundary points of blocks. These observations are characteristic of the interpolation scheme being used, which is known to exhibit larger errors near the boundaries of the interpolation interval (Runge phenomena).

Although the error for schemes with  $n_I \neq n_b^0 + 1$  are not presented, it is readily deduced from our reported results that for any choice of  $n_b^0$  the error can be controlled by changing  $n_I$ . Furthermore, different choices of  $n_{\max}$  were not explored since  $n_{\max} = 10$  allows for schemes with errors as small as  $\sim 10^{-12}$ , which are sufficient for many practical applications. Errors smaller than  $10^{-12}$  can be obtained by reducing  $n_{\max}$  and increasing  $n_I$ .

#### 4.2. Computation time

The test cases used to examine the computation time of the FLGF method follow the same setup as in the previous section, except that we now consider three types of active grid geometries: cubes, spheres, and spherical-shells (with a thickness of 0.1 diameters). For cases of spheres and spherical-shells, the union of blocks that define the active grid are distributed in such that way that they best approximate these geometries. Computation times for various schemes and problem sizes are presented in Fig. 2, and asymptotic computation rates (total number of active grid points / computational time) for a few schemes are included in Table 2.

Table 2: Asymptotic computation rates for selected test cases presented in Fig. 2. Values given in units of  $10^5$  pts/s. Rates are based on the test cases with  $10^8$  active grid points (values interpolated from nearest two data points). Asterisk (\*) indicates rates that are not strictly asymptotic since they correspond to  $\mathcal{O}(N \log N)$  schemes.

scheme	box	sphere	spherical-shell
$n_I = 7$	1.187	1.167	1.267
$n_I = 13$	1.189	1.169	1.315
$n_I = 28$	1.384	1.341	1.150
FFT Conv.	3.540*	n/a	n/a

As expected, the results for all schemes presented in Fig. 2, with the exception of the one labeled “FFT Conv.”, have an asymptotic computational complexity of  $\mathcal{O}(N)$ . “FFT Conv.” refers to the special case where the entire active grid is a single block for which the FLGF method reduces to a single FFT-based convolution. This special case is a useful point of comparison since the algorithmic complexity and efficiency are determined by the highly-optimized third-party library used to compute FFTs. For the grid geometries considered, Table 2 indicates that our implementation of the FLGF method achieves computation rates that are roughly a third of the rate of the single block case (for  $N = 10^8$ ). Furthermore, Fig. 2 demonstrates that the rates of all test cases are within a factor of 10 of those corresponding to the single block case.

A detail report of the computation time of the *Pre-computation* step is omitted, since this step is observed to require only a small fraction of the computation time of a single solve. We substantiate this claim by reporting that for test cases involving cubic active grids containing  $\mathcal{O}(10^1)$ ,  $\mathcal{O}(10^2)$ , and  $\mathcal{O}(10^4)$  blocks the *Pre-computation* step required less than 10%, 1%, and 0.1%, respectively, of the computation time of a single solve. These results are consistent with the fact that the operation count of the pre-computation step increase with the number of levels, which in turn increase logarithmically with the number of blocks for test cases involving cubic active grids.

#### 4.3. Parallel performance

The parallel performance of the FLGF method is investigated by considering cubic active grids and using the scheme corresponding to  $n_I = 13$  described in the previous sections. Computation rates and parallel efficiencies for various problem sizes with core counts between 12 and 660 are included

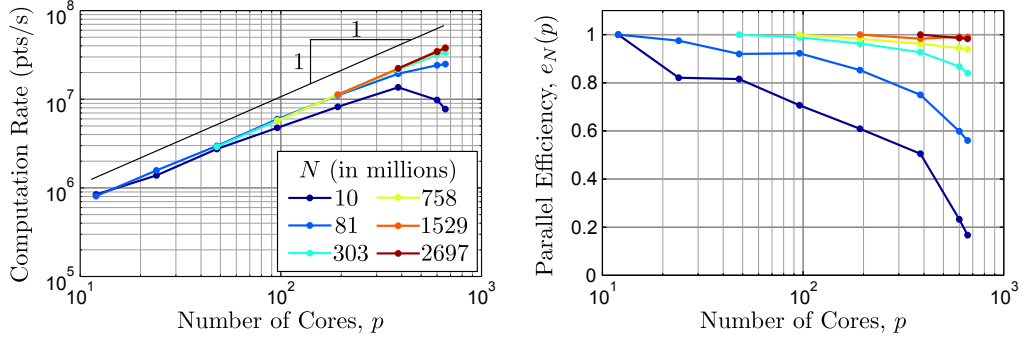


Figure 3: Computation rates (*left*) and parallel efficiencies (*right*) for cubic active grid of various sizes. The parameter  $n_I$  is set to 13 for all test cases. The listed values of  $N$ , in ascending order, correspond to problems containing  $5.8 \times 10^3$ ,  $4.6 \times 10^4$ ,  $1.8 \times 10^5$ ,  $4.4 \times 10^5$ ,  $8.3 \times 10^5$ , and  $1.6 \times 10^6$  active blocks.

in Fig. 3. For all reported test cases the number of cores is a multiple of 12 (there are 12 cores per node in our test machine), and each MPI-process is mapped to a single core. The parallel efficiency for each test series is defined by

$$e_N(p) = \frac{p_{\min}}{p} \frac{T_N(p_{\min})}{T_N(p)}, \quad (36)$$

where  $N$  is the total number of active grid points in the test series,  $p$  is the number of cores,  $p_{\min}$  is the minimal number of cores considered in the test series, and  $T_N(p)$  is the runtime of the test problem.

Both strong and weak scaling can be inferred from the left plot in Fig. 3. Strong scaling, i.e. fixed  $N$  and increasing  $p$ , corresponds to the individual curves associated with each test series. Weak scaling, i.e. fixed  $N/p$  and increasing  $p$ , is achieved when the curves associated with different test series collapse. Figure 3 demonstrates that, over a reasonable range  $p$ , the curves for most of the test series collapse to a single line of approximately unit slope. This indicates that our implementation exhibits both good strong and weak scaling.

There are two main considerations that affect the performance of our parallel implementation. The first is the number of blocks per core. The FLGF method is broken into block-wise operations. If there too few blocks per core our total work cannot be evenly distributed across all cores. Furthermore, given our communication scheme for the *Level Interactions* step, fewer blocks per core is likely to increase the total number MPI messages sent and

received. The second consideration is the amount work per core. If the work per core is too small then communication cost can take an overwhelming fraction of the net runtime. Based on these considerations and the reported results, we conclude that if each core has, on average, more than 300,000 active grid points and 200 blocks then the parallel efficiency is expected to be above 80%. This observation seems consistent with the results reported on other MPI-based implementations of kernel-independent FMMs, for example [30].

In the interest of completeness, we note that computation rates reported in Fig. 3, in particular those corresponding to 12 cores, are roughly half the rates expected based on the our serial results. This decrease in performance due to an increase in cache-misses when more than one core per node is used. We expect that future, higher-quality, implementations of the FLGF method can readily mitigate this feature.

## 5. Conclusions

We have presented a new kernel-independent fast multipole method for elliptic difference equations on infinite Cartesian grids. The FLGF method exploits the regularity of the underlying grid to achieve small computation times by using a fast convolution technique that combines interpolation-based kernel compression and FFTs. Interpolation based on equidistant nodes, along with an  $p$ - and  $h$ -refinement technique, is shown to be an effective scheme for obtaining low-rank representations of kernels, while still preserving sufficient regularity to allow discrete convolutions to be performed quickly using FFTs. The adaptive block-structured grid strategy blends well with the overall algorithm, and the reported numerical experiments demonstrate that computation rates remain roughly invariant of source distributions.

The efficiency of the FLGF method is demonstrated through several numerical experiments solving the discrete 3D Poisson equation for cases involving up to 2 billion grid points and 660 cores. Serial test cases confirm that the algorithm archives an asymptotic linear complexity. Computation rates of approximately  $1.2 \times 10^5$ pts/s or, equivalently, grind times of  $8.3\mu$ s are observed for problems containing  $10^8$  grid points. The computation rate is shown to be roughly invariant to different source distributions and block sizes. Furthermore, the time required to perform all pre-computations for typical problems is shown to negligible. Test cases investigating the parallel performance of our implementation demonstrate that parallel efficiencies higher

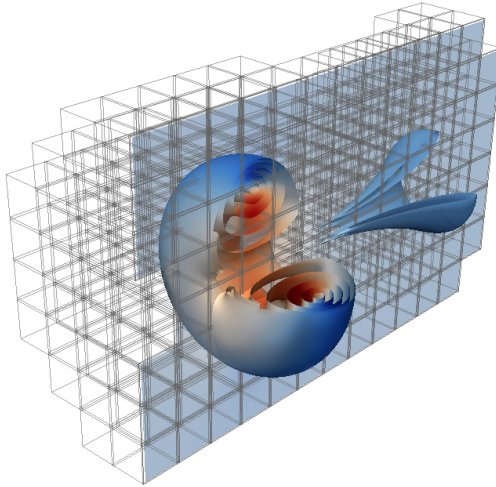


Figure 4: Vortex ring at a Reynolds number of 7,500. Isocontours correspond to the absolute value of vorticity (log scale), color corresponds to the streamwise velocity, and grey boxes correspond to the location of grid blocks used in the simulation.

than 80% are achieved under modest considerations (at least 200 blocks and 300,000 grid points per core).

The FLGF method is particularly useful for solving PDEs that have been discretized using a numerical scheme that enforces discrete conservation laws. In such cases, accurate solutions to the difference equations, but not necessarily the original PDE, are necessary to preserve physical fidelity. We have applied the present method to solve incompressible, viscous, external flows using a finite volume scheme and an infinite staggered Cartesian grid. Figure 4 includes a snapshot of thin vortex ring at a Reynolds number (based on ring circulation) of 7,500 simulated using this scheme. A detailed description and results pertaining to the application of the FLGF method to the incompressible Navier-Stokes is the subject of future publications.

In the interest of brevity, our discussion and reported results only pertain to the discrete Laplace kernel, yet the FLGF method can be applied to other non-oscillatory LGFs. In fact, our method can be readily generalized to any non-oscillatory kernel (including singular kernels); the only restriction is that sources and evaluation points be defined on an regular grid. Based on these observations, and the simple/standard routines and data-structures involved in the algorithm, it is expected that the FLGF method can be readily incorporated into a wide range of existing methods and codes that



solve elliptic PDEs on unbounded domains.

## Appendix A. Numerical evaluation of the fundamental solution of the Laplace operator

Values of  $\mathbf{G}(\mathbf{n})$  for small  $|\mathbf{n}|$  are frequently used by the FLGF method. Therefore it is advantageous to program an accurate look-up table for values of  $\mathbf{n}$  confined to a small cubic box centered at the origin. The symmetry of  $\mathbf{G}(\mathbf{n})$ , discussed in Section 2.1, suggests that only approximately 1/48-th of the total number of points in the box need to be numerically evaluated. It is possible to reduce the triple integral in Eq. 6 to a single semi-infinite integral [31] given by

$$\mathbf{G}(\mathbf{n}) = - \int_0^\infty e^{-6t} I_{n_1}(2t) I_{n_2}(2t) I_{n_3}(2t) dt, \quad (\text{A.1})$$

where  $I_k(x)$  is the modified Bessel function of the first kind of order  $k$ , and  $\mathbf{n} = (n_1, n_2, n_3) \in \mathbb{Z}^3$ . In our experience, it is easier and faster to numerically integrate Eq. A.1 instead of Eq. 6. The integrand of Eq. A.1 is non-oscillatory and smooth throughout the domain of integration. A simple adaptive Gauss-Kronrod scheme can be used to perform the numerical integration and obtain error estimates. Furthermore, the semi-infinite integral can be partitioned into two intervals  $[0, \alpha]$  and  $[\alpha, \infty]$ , where  $\alpha$  is chosen such that the latter integral can be evaluated analytically using the asymptotic expansion (for large arguments) of  $I_n(x)$  [32]. More efficient implementations might consider partitioning the integration interval into multiple subintervals, exploiting both the ascending series representation and the asymptotic expansion of each Bessel function.

## References

- [1] Z. Gimbutas, V. Rokhlin, A generalized fast multipole method for nonoscillatory kernels, *SIAM J. Sci. Comput.* 24 (2003) 796–817.
- [2] L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, *J. Comput. Phys.* 196 (2004) 591–626.
- [3] P. G. Martinsson, V. Rokhlin, An accelerated kernel-independent fast multipole method in one dimension, *SIAM J. Sci. Comput.* 29 (2007) 1160–1178.

- [4] W. Fong, E. Darve, The black-box fast multipole method, *J. Comput. Phys.* 228 (2009) 8712–8725.
- [5] B. Zhang, J. Huang, N. P. Pitsianis, X. Sun, A Fourier-series-based kernel-independent fast multipole method, *J. Comput. Phys.* 230 (2011) 5807–5821.
- [6] A. Gillman, P. G. Martinsson, A fast solver for poisson problems on infinite regular lattices, *Journal of Computational and Applied Mathematics* 258 (2014) 42–56.
- [7] O. Buneman, Analytic inversion of the five-point Poisson operator, *J. Comput. Phys.* 8 (1971) 500–505.
- [8] P. G. Martinsson, G. J. Rodin, Asymptotic expansions of lattice Green’s functions, *Proc. Roy. Soc. Lond. A* 458 (2002) 2609–2622.
- [9] A. Gillman, P. G. Martinsson, Fast and accurate numerical methods for solving elliptic difference equations defined on lattices, *J. Comput. Phys.* 229 (2010) 9026–9041.
- [10] W. H. McCrea, F. J. W. Whipple, Random paths in two and three dimensions, *Proc. Roy. Soc. Edinburgh* 60 (1940) 281–298.
- [11] E. W. Montroll, R. B. Potts, Effect of defects on lattice vibrations, *Phys. Rev.* 100 (1955) 525–543.
- [12] E. N. Economou, *Green’s functions in quantum physics*, Springer-Verlag, Berlin, 1984.
- [13] J. H. Bramble, B. E. Hubbard, On the formulation of finite difference analogues of the Dirichlet problem for Poisson’s equation, *Numer. Math.* 4 (1962) 313–327.
- [14] P. G. Martinsson, G. J. Rodin, Boundary algebraic equations for lattice problems, *Proc. Roy. Soc. Lond. A* (2009).
- [15] J. B. Perot, Discrete conservation properties of unstructured mesh schemes, *Annu. Rev. Fluid Mech.* 43 (2011) 299–318.

- [16] D. N. Arnold, P. B. Bochev, R. B. Lehoucq, R. A. Nicolaides, M. Shashkov, Compatible spatial discretizations, volume 142 of *The IMA Volumes in Mathematics and its Applications*, Springer New York, New York, 2006.
- [17] M. L. Glasser, I. J. Zucker, Extended Watson integrals for the cubic lattices, *Proc. Natl. Acad. Sci. USA* 74 (1977) 1800–1801.
- [18] R. T. Delves, G. S. Joyce, On the Green function for the anisotropic simple cubic lattice, *Ann. Phys.* 291 (2001) 71–133.
- [19] R. J. Duffin, E. P. Shelly, Difference equations of polyharmonic type, *Duke Math. J.* 25 (1958) 209–238.
- [20] M. Mangad, Asymptotic expansions of Fourier transforms and discrete polyharmonic Greens functions, *Pacific J. Math* 20 (1967) 85–98.
- [21] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, *J. Comput. Phys.* 73 (1987) 325–348.
- [22] L. Greengard, V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, *Acta Numer.* 6 (1997) 229–269.
- [23] A. Dutt, M. Gu, V. Rokhlin, Fast algorithms for polynomial interpolation, integration, and differentiation, *SIAM J. Numer. Anal.* 33 (1996) 1689–1711.
- [24] C. Berman, Grid-multipole calculations, *SIAM J. Sci. Comput.* 16 (1995) 1082–1091.
- [25] J. R. Phillips, J. K. White, A precorrected-FFT method for electrostatic analysis of complicated 3-d structures, *IEEE Trans. Comput. Aid. D.* 16 (1997) 1059–1072.
- [26] L. N. Trefethen, *Spectral methods in MATLAB*, volume 10, SIAM, Philadelphia, PA, 2000.
- [27] A. Quarteroni, F. Saleri, P. Gervasio, *Scientific computing with MATLAB and Octave*, Springer, Berlin, 2010.

- [28] S. G. Johnson, M. Frigo, A modified split-radix FFT with fewer arithmetic operations, *IEEE Trans. Signal Process.* 55 (2007) 111–119.
- [29] G. M. Morton, A computer oriented geodetic data base and a new technique in file sequencing, IBM (1966).
- [30] L. Ying, G. Biros, D. Zorin, H. Langston, A new parallel kernel-independent fast multipole method, in: *Proceedings of the 2003 ACM/IEEE Conference on Supercomputing*, 2003, pp. 14–14.
- [31] J. Cserti, Application of the lattice Greens function for calculating the resistance of an infinite network of resistors, *Am. J. Phys.* 68 (2000) 896.
- [32] M. Abramowitz, I. A. Stegun, *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, Dover, New York, 1972.